TP1 : Optimisation

Lors de ce TP, nous allons voir différentes méthodes ayant pour but de reconstruire une image à partir d’une image bruitée. On note la représentation vectorielle de l’image originale et celle de l’image dégradée par un bruit (et un opérateur flou (. Nous etudierons tout d’abord les images ainsi que le bruit en traçant son histogramme. Puis, nous procéderons à la reconstruction de l’image avec la méthode de Tikhonov et ensuite avec la méthode de descente de gradient.

 Lorsque nous affichons les images originales et dégradées nous obtenons les images suivantes :

Ces deux images sont de dimension 256x256. Les images sont données dans des *struct* il faut donc utiliser la commande *reshape* de Matlab afin de reconstruire les images en dimension 256x256.

On s’interesse maintenant à la réponse fréquentielle du filtre. Pour isoler le bruit , on soustrait à l’image dégradée la représentation vectorielle de l’image originale floutée. On obtient donc

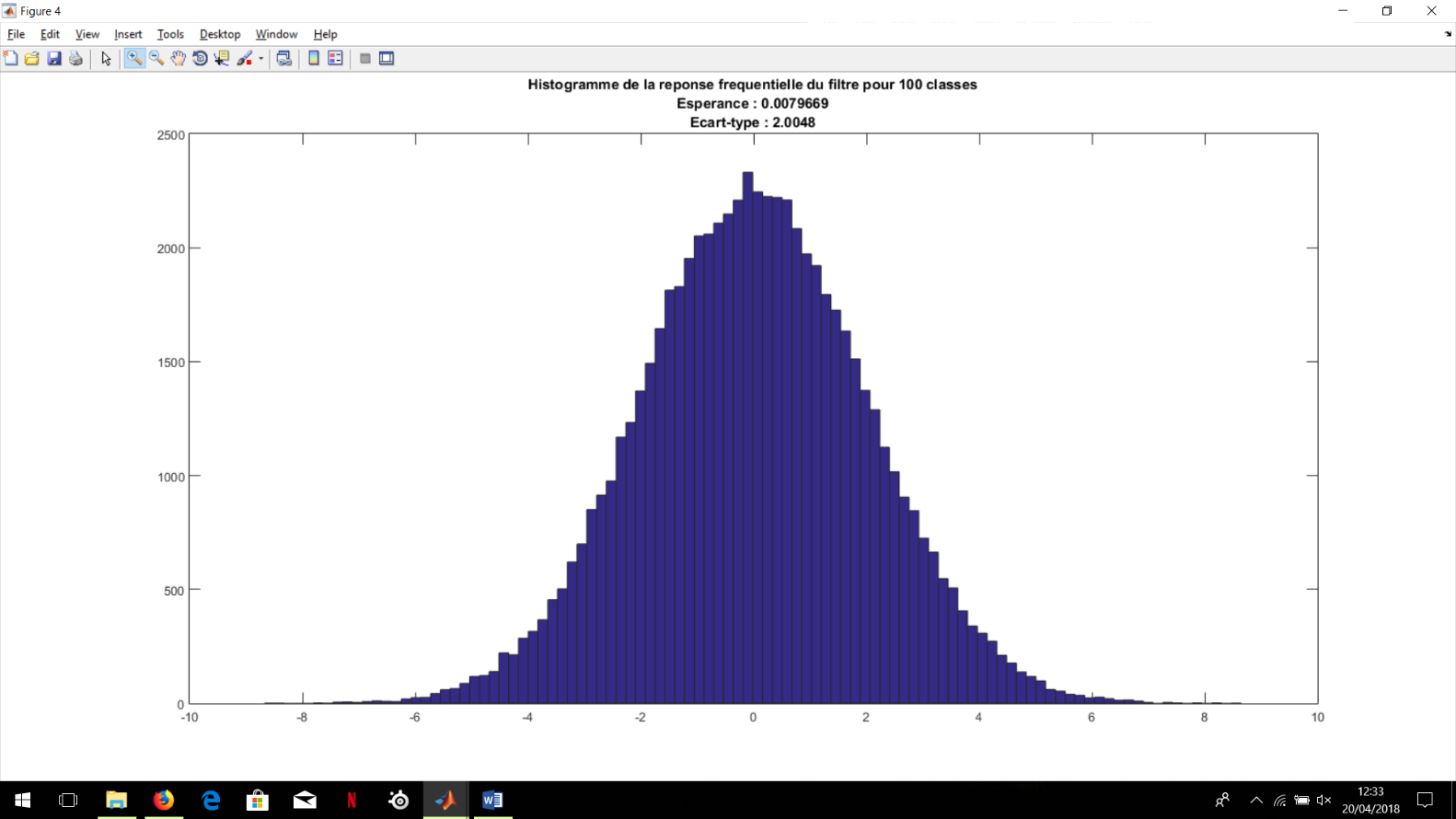
.

Lors de cette opération nous devons faire attention à bien faire la transformée de Fourier inverse dans l’opération afin de ne pas rester dans le domaine fréquentiel.

Le code Matlab associé donne :

err= I3image - real(ifft2(fft2(I1.x).\*I4.H));

Pour vérifier que le bruit est bien centré, nous allons tracer son histogramme et calculer son espérance.



On remarque en calculant avec la fonction *hist* de Matlab que l’histogramme suit bien une distribution gaussienne. De plus celle-ci est centrée, son espérance est presque nulle, et sont écart-type vaut 2.

Maintenant nous allons nous interesser à la recontruction avec méthode de Tikhonov. On va chercher à produire une image restaurée tel que .

vérifie l’équation ci-dessus si . Ce qui s’écrit :

Si nous écrivons cela sous Matlab nous obtenons le code suivant :

%% Question 3

% Reconstitution de l'image d'origine

lambda=0.5;

I4H=I4.H; %image du flou

ILG=Ilap.G;

Ireconstruit = real(ifft2((1./(conj(I4H).\*I4H + lambda\*conj(ILG).\*ILG )).\*(conj(I4H)) .\*fft2(I3image)));

figure();

subplot(1,2,1)

imagesc(Ireconstruit);

colormap('gray');

title({['\bfReconstruction par la methode'], ['de Tikhonov pour \lambda=',num2str(lambda)]});

subplot(1,2,2)

imagesc(I3image);

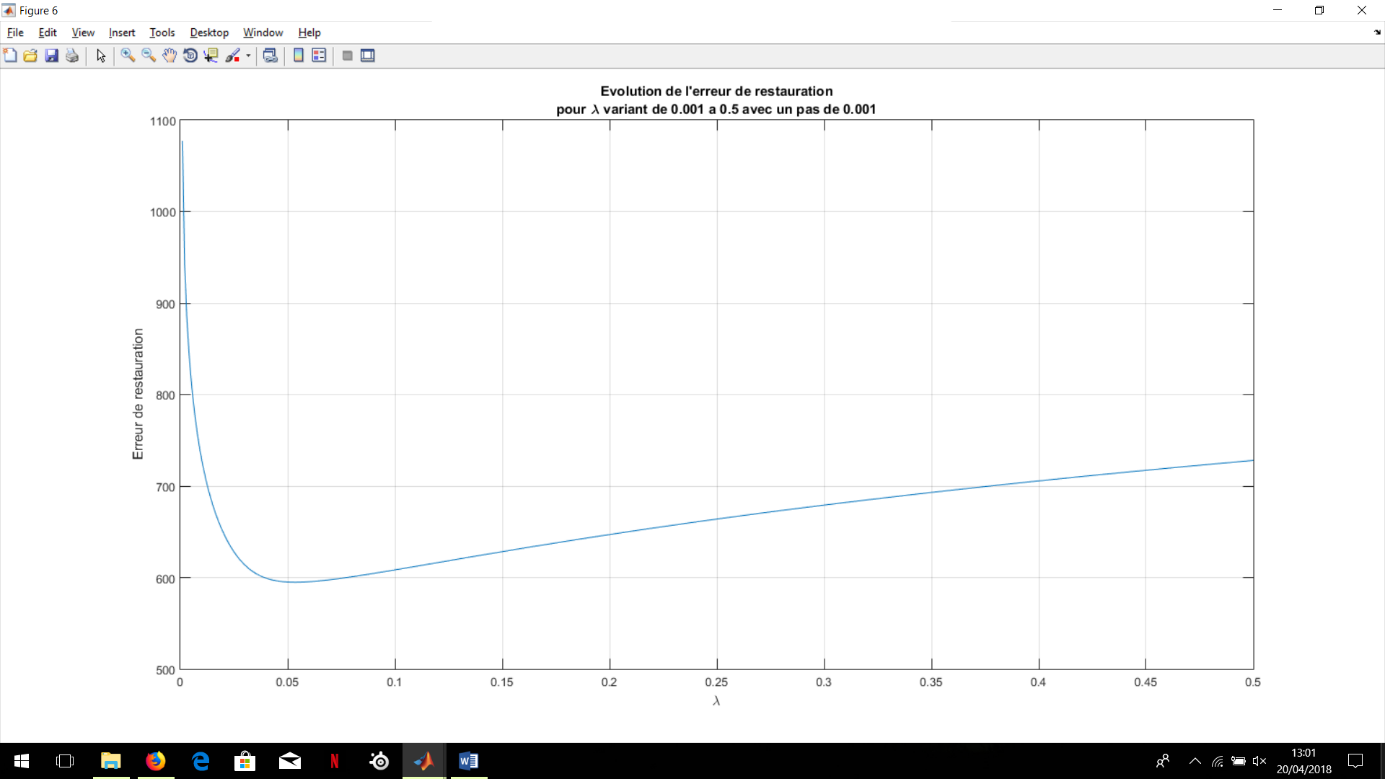
title('\bfImage degradee');

Et nous obtenons la figure ci-dessous :



Il faut bien faire attention comme précedemment à faire les calculs dans le domaine de Fourier et repasser ensuite dans le domaine temporel.

On peut aussi tracer l’évolution de l’erreur entre l’image reconstruite et l’image dégradée en fonction de pour determiner la valeur du optimal.



Méthode de la descente du gradient :

On cherche maintenant à résoudre , étant un opérateur de flou, D un opérateur de décimation et .



Pour reconstruire cette image, il est nécessaire d’utiliser la méthode de descente de gradient. En effet, on peut observer des « trous » dans l’image dû à l’opérateur de décimation. Ainsi, il n’est pas possible de savoir comment ils seront traités dans le domaine de Fourier. Par conséquent, l’utilisation de la méthode de Tikhonov est inadaptée.

On utilise alors la méthode de descente de gradient pour reconstruire cette image.

Cette méthode consiste à minimiser le gradient par itérations successives. On cherche alors à déterminer une suite convergeant vers l’image originale qui possède un gradient nul.

Pour ce faire, on définit la suite avec le pas et la direction. La direction s’effectue en fonction de la possibilité de trouver vérifiant la condition .

On réalise alors un développement de Taylor :

Ainsi, pour vérifier la dernière condition, d’où .

La suite s’écrit donc : .

Or de gradient β-lipschitz, donc afin d’obtenir une suite convergente, on prend .

On a, dans notre cas, .

Finalement, l’évolution de la norme du gradient est la suivante :



Code associé à l’utilisation du masque de décimation:

cameramanCS=load('cameraman\_cs.mat');

Decimation=load('decimation.mat');

D=Decimation.D;

CS\_R = cameramanCS.y;

D = reshape(D,256,256);

figure();

imagesc(D);

colormap('gray');

title('decimation');

tI=256;

Z=zeros(tI,tI);

ind=find(D==1);

Z(ind)=CS\_R;

Z=reshape(Z,tI,tI);

figure();

imagesc(Z);

colormap('gray');

title('Image avec decimÃ©e');

x\_nD= I3image;

for k=0:1:n

x\_nD=x\_nD-Gamma.\*(conj(Z).\*(Z.\*x\_n-yI) +(lambda .\*conj(ILG).\*ILG.\*x\_nD));

end

figure()

imagesc(real(ifft2(x\_nD))),colormap('gray');

title('Image x\_nD');